Dynamically Partitioning Workflow over Federated Clouds For Optimising the Monetary Cost and Handling Run-Time Failures

Dynamically Partitioning Workflow over Federated Clouds For Optimising the Monetary Cost and Handling Run-Time Failures

Zhenyu Wen*§, Rawaa Qasha*‡, Zequn Li†, Rajiv Ranjan* ¶, Paul Watson*, Alexander Romanovsky*

Abstract—Several real-world problems in domain of healthcare, large scale scientific simulations, and manufacturing are organised as workflow applications. Efficiently managing workflow applications on the Cloud computing data-centres is challenging due to the following problems: (i) they need to perform computation over sensitive data (e.g. Healthcare workflows) hence leading to additional security and legal risks especially considering public cloud environments and (ii) the dynamism of the cloud environment can lead to several run-time problems such as data loss and abnormal termination of workflow task due to failures of computing, storage, and network services. To tackle above challenges, this paper proposes a novel workflow management framework call DoFCF (Deploy on Federated Cloud Framework) that can dynamically partition scientific workflows across federated cloud (public/private) data-centres for minimising the financial cost, adhering to security requirements, while gracefully handling run-time failures. The framework is validated in cloud simulation tool (CloudSim) as well as in a realistic workflow-based cloud platform (e-Science Central). The results showed that our approach is practical and is successful in meeting users security requirements and reduces overall cost, and dynamically adapts to the run-time failures.

Index Terms—Cloud Federation, Scientific Workflow Optimisation, Deployment, Security, Monetary Cost, Scheduling.

1 Introduction

Scientific workflows have become an increasingly popular paradigm for enabling and accelerating scientific data analysis. They consist of a series of computational tasks that are logically connected by data and controlling flow dependencies. They have been successfully run on traditional HPC (High Performance Computing) systems and clusters. However, in recent years many researchers have migrated workflow systems onto the cloud in order to exploit the economic and technical benefits of this technology [1].

The Cloud computing provides a computing paradigm that focuses on the on-demand provisioning of computing resources, including hardware, software, storage and network. Furthermore, cloud providers have distributed several data centres at different geographical locations over the internet in order to deliver quality services for their customers around the world [2] [3]. The currently available cloud platforms distinguish themselves on service type, the cost, the Quality of Service (QoS) as well as performance [4]. This fact enables cloud customers to freely select their target architecture from a broad range of cloud platforms.

* *The authors are with the School of Computing Science, Newcastle University, United Kingdom. E-mail:{z.wen,r.qasha,Raj.Ranjan,paul.watson,alexander.romanovsky}@newcastle.ac.uk,*

† *The author is with the school of mathematics and information science, Northumbria university, United Kingdom. E-mail:{Zequn.li}@northumbria.ac.uk,*

‡ *The author is with the College of Computer Sciences and Mathematics, Mosul university, Iraq.*

§ *The author is with the School of Informatics, University of Edinburgh, United Kingdom. E-mail:{zwen}@inf.ed.ac.uk*

¶ *The author is with the School of Computer, Chinese University of Geosciences, Wuhan, China*

Manuscript received ???; revised ???

Although users are interested in the execution of their workflow applications in the cloud, the current implementations mainly address a single cloud. In addition, existing systems are unable to coordinate across different cloud-based data centres in order to optimally allocate application services to meet users' functional (i.e. hardware) or non-functional (i.e. security) requirements [5]. Further, cloud providers are subject to failures. For example, an outage at GoDaddy took down millions of web sites [6] and a 12 hour Amazon EC2 outage [7] raised serious questions about reliability on a single cloud provider.

Considering these issues, cloud federation has the potential to facilitate just-in-time, opportunistic, and scalable provisioning of application services, consistently fulfilling user requirements under variable workload, resource and network conditions [8]. Using a federated cloud, users are able to deploy their applications over a set of clouds from different cloud providers across different geographical locations, bringing various advantages such as leveraging unique cloud specific services, higher availability and redundancy, disaster recovery and geo-presences.

Motivated by the above considerations, we propose to deploy scientific workflow over utility-oriented cloud federations, in the meantime ensuring that the deployment can meet users' security requirements and minimise monetary cost, while handling changes in cloud availability (including the existing cloud virtual machines (VMs thereafter) outage and the new VMs becoming available).

We assume that there is a set of workflow execution environments running on different data centres which are owned by different cloud providers. Further, the different computing resources have different security levels, for example, Windows Azure provides private cloud and public

cloud that come with different security levels. Therefore, the security of a scientific workflow can be improved by deploying sensitive services or data to more secure clouds. Likewise, the cost can be reduced through distributing the less sensitive services or data to cheaper clouds with lower security levels. In addition, we allow for the cloud federation to be very dynamic, as the availability of clouds may sometimes change.

According to the above assumptions, the deployment of scientific workflow on a federated clouds poses a number of challenges:

- The considerable amount of computation required for exploring the optimal deployment. We assume a workflow with S tasks can be deployed over a federated cloud that includes C clouds. Therefore the total number of deployments is C^S which is exponential to the number of workflow tasks.
- Tasks in the workflow system and their corresponding security levels are influenced by different aspects, such as user preference, the task requirements and inputs/outputs data.
- The cost of the deployment is dependent on several factors, including data storage cost, data communication cost and computation cost.
- The trade-off between secure deployment and monetary cost is also a challenge.
- Dynamic handling of cloud environment changes. This requires to rapidly generate new deployment solutions when the cloud environments change.

Few of works have been done to address deploying workflow over federated cloud. [9] introduces a static algorithm to deploy workflows over federated clouds that meets security requirements. However, the dynamic of cloud environments is not considered. In our previous work [10], we have considered cloud failure during workflow running, while meeting the security requirements and minimising the cost. However, the proposed method cannot handle the large scale workflow and fails to generate a better solution when a new cloud joins the cloud federation.

In this paper, we propose DoFCF (Deploy on Federated Cloud Framework), a framework for deploying workflows over federated clouds that meets security requirements, and optimising the monetary cost, while dynamically handling the availability change of federated clouds.

Our framework provides a set of solutions for workflow scheduling, including where to deploy tasks, when to start each service and how to handle the cloud availability change. The deployment of the workflow over federated cloud is based on adhering to a set of specific security requirements and minimisation procedures. Additionally, DoFCF offers a dynamic solution to dynamically reschedule the running workflow to new clouds, in order to complete the execution of an unfinished workflow or save on costs.

1.1 Paper Contributions

Considering the above challenges and problems, this paper makes the following core contributions:

- A framework to model the security constraint of workflow deployment and the situation of cloud availability change during the workflow execution time. The framework also quantifies the cost of executing workflow over federated clouds.
- Investigation of the existing state-of-the-art optimisation algorithms. Further, we extend two classic algorithms and adapt to DoFCF to achieve rapid exploration for a possible deployment solution. In order to handle the availability change of cloud resources, a novel dynamic rescheduling algorithm is developed to resume workflow execution when failures occur or reduce the monetary cost by redeploying the running workflow to cheaper clouds.
- Evaluating the implemented framework on CloudSim [11] which is a Cloud simulator and e-Science Central [12] (e-SC), a real scientific workflow based cloud platform.

The rest of this paper is organised as follows. In Section 2 the basic models of the framework are discussed. Next, a specific security model is adapted to DoFCF, demonstrating how to deploy a workflow over a federated cloud to meet security requirements while minimising the cost. Then the state-of-the-art optimisation algorithms are explored, extended and adapted to our DoFCF to optimise workflow partitioning. In Section 5, we evaluate the framework by using CloudSim, and also develop a tool to schedule the workflows over a set of e-SC instances. Before drawing conclusions in Section 7, we discuss the related work.

2 Basic Modelling Constructs

In this section, we present a system model of deploying workflow applications over federated clouds. In the following, the general scientific workflow model and security model are introduced. In addition, a general cost model will be used to calculate the monetary cost of deployment. Moreover, we present an optimisation model that can guarantee the deployment solution meets the security constraints as well as minimising the cost. Finally, a dynamic cost model is used to help rescheduling the running workflow when the cloud availability change in a federated clouds. Table 1 shows the notations for the rest of the paper.

2.1 System model

A Cloud Service Broker performs cloud exchange and negotiates with each available cloud to allocate resources that meet user' specific requirements. In this paper, we propose a Cloud Service Broker which can partition workflow applications over federated clouds. Fig 1 shows the architecture of the Cloud Service Broker for workflow application along with other components as illustrated below:

The *Client* can be a platform for workflow management such as e-Science Central or Pegasus [13] which allows users to describe and submit their workflow application through platform components. The **Workflow Engine** delivers the workflow tasks (or services in this paper) to the underlying Cloud Service Broker, including execution requirements, task description, and the desired security requirements.

The *Cloud Service Broker* enables the functions of resource allocation, workflow scheduling and software deployment.

Symbol	Meaning
Workflow	
s_i	ith service in a workflow application
$d_{i,j}$	data dependency between s_i and s_j
$\mathcal{O}$	the union of data dependencies and services
o	one element of set $\mathcal{O}$
c_i	ith cloud of a set of clouds C
Λ	possible deployment solutions
λ	one of the deployments of Λ
$Selected$	set of the services that need to be rescheduled
$Input$	set of data that have already been generated
Λ'	possible deployments of the services in $Selected$
λ'	one of the possible deployments of Λ'
Cost Model	
s_i^c	service s_i is deployed on cloud c
$T_{i,j}$	storage time of data $d_{i,j}$
$Store_c$	cost of storing data on cloud c in GB per hour
OUT	the outgoing data dependencies of a cloud
$Com_{c',c}$	cost of transferring 1GB of data between clouds
IN	the incoming data dependencies of a cloud
T_j^c	execution time of s_j on cloud c
$Exec_c$	cost of using compute resources on c for one hour
$Scost$	data storage cost
$Ccost$	communication cost
$Ecost$	execution cost
$Icost$	initial cost for setting up a new deployment
$Dcost$	cost of the new deployment
$COST$	total cost of a workflow deployment
Security Model	
$func1$	embeds constraints for each o
$func2$	represents the constraints for the whole workflow
NWD	no-write-down
NRU	no-read-up
SIC	security in cloud computing

TABLE 1: Notations

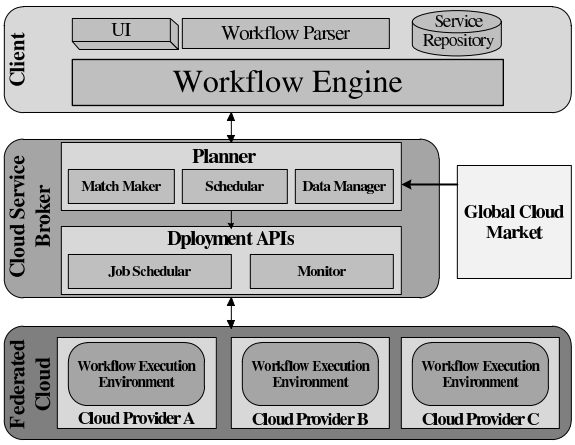


Fig. 1: Architecture of Cloud Service Broker for Scientific Workflow

Our framework includes a **Planner** component that performs a matching process to select the target clouds for deployment, based on the information passed from **Global Cloud Market**. Further, the workflow tasks are assigned by the **Scheduler**, and the **Data Manager** maintains the data transfer during workflow execution. The planned tasks are distributed to the underlying cloud providers via **Deployment APIs**. These APIs can also be used to interact with the underlying clouds to monitor workflow execution and cloud availability.

Federated cloud is a cloud resource pool that provisions computation and storage resources, as well as specific non-functional capabilities (referring to different security levels in this paper) and functional capabilities such as the execution environment of each tasks. For example, Jclouds [14] provides the API to use portable abstractions or cloud-specific features.

2.2 Scientific Workflow

A workflow-based application consists of a set of services and data. It is modelled as a Directed Acyclic Graph (DAG), $G = (S, E)$, where S is the set of services, and E is a set of dependencies between those services. Services are represented by the graph vertices and the edges represent the dependencies between those services. Although a workflow-based application can have several different types of dependency relationships, in this work we only consider the data dependency (this is the most common dependency relationship in scientific workflow applications). In this type of dependency, a data item is generated from a source service and consumed by a destination service. For example, $e_{i,j}$ represents a data dependency between service s_i and service s_j. To represent data dependencies we use a distance matrix $D = [d_{i,j}]$ of size $|S| \times |S|$ where a positive value of $d_{i,j}$ indicates a dependency between s_i and s_j as well as the size of transmitted data. $\mathcal{O}$ represents the union of D and S. Furthermore, C represents a set of clouds which are available for deployment.

2.3 General Security Model

An application's security can be improved by two approaches: firstly, refining the design and implementation of the application; secondly, deploying the application over more trustworthy resources, such as shifting the application to a higher security server. In this paper, we propose to increase the security of a workflow by adopting the latter approach. To achieve the enhancement in workflow security, we present two functions which are used to provide a concrete representation for different types of security requirements. We assume that Λ represents the possible deployment solutions for given workflow over federated clouds C and λ is one deployment of Λ, noting $\Lambda = \mathcal{O} \times C$ and $\lambda \in \Lambda$.

func1) embeds constraints for d and s. Thus, if λ is a valid deployment solution, each $o \in \mathcal{O}$ has its security constraints and must be deployed on a cloud $c \in C$ which can meet the constraint.

func2) represents the constraints for the whole workflow deployment. Therefore, a valid deployment solution λ must meet the security constraint H. Where H is one of the security requirements.

2.4 Cost Model

The cost model is designed to calculate the cost of deploying a workflow over a set of available clouds, including data storage cost, data communication cost and computation cost. We assume that the clouds are linked in a fully connected topology and the data can be transferred between clouds without obstructions. Additionally, a cloud c can run several services s at the same time. Therefore, a set of cost functions is defined as follows:

The first function is the data storage cost:

$$Scost(s_i^c) = \sum_{d_{i,j} \in OUT} d_{i,j} \times T_{i,j} \times Store_c \quad (1)$$

Where s_i^c means that service s_i is deployed on cloud c. OUT is a set of data dependencies, representing the data that are generated by s_i and transferred to its immediate successor s_j which is not deployed on c (note that if all immediate successors of s_i are on c, then $OUT = \emptyset$). $d_{i,j}$ represents the amount of data which is generated by s_i and consumed by s_j. $T_{i,j}$ denotes storage time of data $d_{i,j}$, which is the required time starting from the generation of data until the completion of workflow execution. Finally, $Store_c$ is the cost of storing 1GB of data for one hour on cloud c.

In this model, we make an assumption that the data remains stored only on the source cloud to avoid double-accounting for the cost. The reason for storing the outputs of a service even after the generated data has been sent to another cloud is to handle a failure of the destination cloud. In this case, the stored data provides a way to resume the computation on another cloud without the need to restart the whole workflow execution. This can be adapted to handle the cloud change problem.

The second function, $Ccost$, is used to estimate the communication cost of transferring data between different services.

$$Ccost(s_i^c) = \sum_{d_{i,j} \in IN} d_{i,j} \times Com_{c',c} \quad (2)$$

It is the data transferred from the immediate predecessors of service s_j (denoted as IN), which are not in the same cloud. $Com_{c',c}$ represents the unit cost of transferring 1GB of data from cloud c' to c. However, if two services are deployed on the same cloud, the cost is zero, i.e. $\forall c' = c : Com_{c',c} = 0$.

Finally, $Ecost(s_i^c)$ indicates the execution cost of service s_i on c. It is defined as:

$$Ecost(s_i^c) = T_i^c \times Exec_c \quad (3)$$

Where T_i^c is the execution time of s_i on cloud c, and $Exec_c$ represents the cost of using compute resources on c for one hour.

Based on the three cost functions, we can formulate the $COST(\lambda)$ function to define the total cost of a workflow deployment over a set of clouds:

$$COST(\lambda) = \sum_{s_i^c \in \lambda} (Scost(s_i^c) + Ccost(s_i^c) + Ecost(s_i^c)) \quad (4)$$

2.5 Deployment Optimisation

As mentioned previously, we propose a model for optimising monetary cost as well as meeting the security requirements. Therefore, the optimisation problem is to find a deployment $\lambda \in \Lambda$ with two constraints: i) the deployment λ must meet the security requirements by belonging to either $func1$ or $func2$. ii) the value of $COST(\lambda)$ should be minimised to obtain deployments with a low cost of execution. We express this problem as:

$$\begin{aligned} &\textbf{minimise}\,(COST(\lambda)) \\ &\textbf{subject to}\,\forall\, o^c \in \lambda :\ func1(o^c) := true \ \text{ OR} \\ &\qquad\qquad func2(\lambda) := true \\ &\qquad\qquad \exists\, \lambda \in \Lambda \end{aligned}$$

In the following, we use $func1$ as an example to prove that the optimisation problem is a NP-complete problem.

Theorem: The optimisation is a NP-complete problem.

Proof: we first verify that the problem of deploying a workflow over a set of clouds to meet security requirements is a NP problem (noting $\exists\, \lambda \in \sum(\Lambda, \mathcal{W})$, where $\mathcal{W}$ represents the security requirements).

The NP-completeness of optimising the cost can be illustrated as follows: we start by transforming PARTITION [15] (one of six core NP-complete problem) to our problem. Let the instance of PARTITION be a finite Set $A = (a_1 ... a_m)$ and a weight $w(a_i)$. We want to have two disjoint subsets A_1 and A_2; $A_1, A_2 \subseteq A$, where $A_1 \cup A_2 = A$ and $A_1 \cap A_2 = \emptyset$, such that $\sum_{a \in A1} w(a) = \sum_{a \in A2} w(a)$.

In order to reduce our problem to a PARTITION problem, we assume that a workflow has m numbers of $\mathcal{O}$, and two clouds are available for deployment. Further, we do not consider the security issue, which means any o can be deployed over any of the two clouds. Therefore, we can have two sets of deployments $C_1 = (o_1 .... o_m)$ and $C_2 = (o_1 .... o_m)$ over the two available clouds

Regarding our problem, we need to have two disjoint subsets C_1' and C_2', where $C_1' \cup C_2' = \mathcal{O}$ and $C_1' \cap C_2' = \emptyset$. This match the conditions of the PARTITION problem. Furthermore, $w(o)$ represents the cost of deploying o onto the cloud, so $\sum_{o \in C_1'} w(o)$ is the cost of set C_1'. However, the PARTITION problem is to find two disjoint sets with the same weight, which has the same complexity as our problem that is trying to find two disjoint sets while minimising the total cost, noting $min(\sum_{o \in C1'} w(o) + \sum_{o \in C2'} w(o))$.

2.6 Dynamic Cost Model

Dynamic Cloud resources may affect workflow execution. Situations arise when individual nodes may fail during the execution, or in some extreme cases the whole cloud is unreachable for several hours. As a consequence of their failure, workflow applications may not be executed to completion. Furthermore, new clouds, possibly attractive because they are cheaper or more secure etc., may become available during the execution of workflow applications. Therefore, to deal with the dynamism of cloud resources, we develop a new cost model that dynamically calculates the cost of deploying uncompleted services over the current available clouds.

We assume a set $Selected$ is composed of the services that need to be rescheduled, including unfinished services as well as the services that have been completely processed, and their outputs are the inputs of unfinished services, but the outputs have not been stored because of the failure of the clouds. The details will be illustrated in Section 4.4.

$Input$ is a set of data which have been already generated from the processed services and required for services in $Selected$, and stored in the available clouds. Based on the definition, we can have the initial cost for setting up a new

deployment, which is the cost of storing the input data of $Selected$. It is defined as:

$$Icost(Selected) = \sum_{d_{i,j} \in Input} d_{i,j} \times T_{i,j} \times Store_c \quad (5)$$

In addition, Λ' is the set of possible deployments of the services in $Selected$ over the available clouds C. Consequently, the cost of the new deployment can be defined as:

$$Dcost(\lambda') = COST(\lambda') + Icost(Selected) \quad (6)$$

Where $\lambda' \in \Lambda'$ represents one of the possible deployments for the services in $Selected$.

3 Secure Deployment

In this section, we apply our previous work [16], based on the Bell-LaPadula [17] Multi-Level Security model [18], to demonstrate how to adapt the security model to DoFCF. This incorporates the security levels of the clouds, data and services to achieve a secure deployment for the workflow over a federated cloud.

In our security model, each service S has two security levels: "Clearance" and "Location". "Clearance" represents the services' highest security level, and "Location" is the required operation security level of the service in a specific application. The data D and cloud C only have "Location". $l(o)$ and $c(o)$ represent the security of location of o and the clearance of o respectively.

W represents the security constants, including three rules:

- ***NWD*** "no-write-down": denotes that a service cannot write data which has a lower security level (required security level) than its own. This can be formalised as: $NWD(d_{i,j}, s_j) = c(s_j) \geq l(d_{i,j})\ ?\ true : false$
- ***NRU*** "no-read-up": means a service cannot read data if the data's location security is higher than the service's clearance security, noting: $NRU(s_i, d_{i,j}) = l(d_{i,j}) \geq l(s_i)\ ?\ true : false$
- ***SIC*** "security in cloud computing" (SIC): defines the location security level of a cloud that should be greater than or equal to the location security level of any service or data that are hosted on this cloud– $SIC(d^c_{i,j}, s^c_j) = l(c) \geq l(s_i) \& l(c) \geq l(d_{i,j})\ ?\ true : false$. Where $d^c_{i,j}$ and s^c_j represent data $d_{i,j}$ and service s_i to be deployed on cloud c.

In APPENDIX, available in the online supplemental material, we run a deployment example to show how to apply the above security rules to a workflow application.

4 Deployment Optimisation Algorithms

In this section, we investigate and analyse some state-of-the-art optimisation algorithms and then extend the Genetic Algorithm (named adaGA) and Greedy Algorithm (named NCF) to adapt to our framework. The architecture of our new framework DoFCF is depicted in Figure 2.

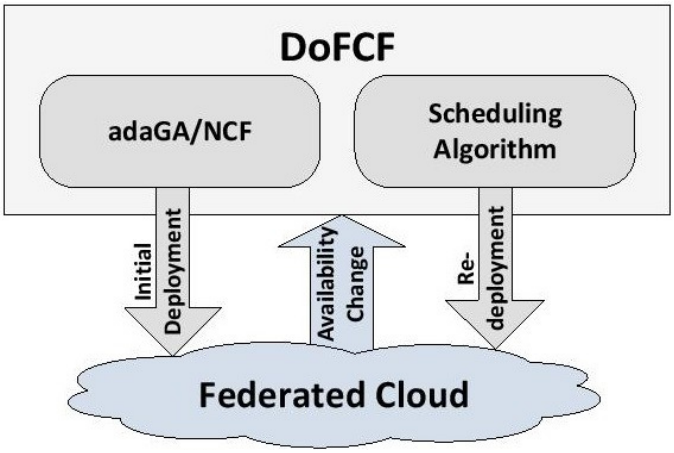


Fig. 2: The Architecture of DoFCF

4.1 Branch and Bound Algorithm

As discussed above, we need to find a $\lambda \in \Lambda$ which minimises the deployment cost. The most common approach is B&B (branch and bound algorithm) [19]. Generally, this type of algorithms require ranking all of the secure deployment solutions and then choose the cheapest one. However, we have proved our problem is a NP-Complete problem, therefore it is very difficult to design an algorithm using B&B in polynomial time as a generic framework.

Although this method gives the optimal solution and guarantees that the result is the cheapest deployment, it is not very scalable. In our paper [20], we demonstrated that when the number of services increased to 12, a version of B&B that we have implemented, required approximately 15 minutes to generate a solution. Thus, this type of algorithms are not considered in our framework.

4.2 Genetic Algorithm

A Genetic Algorithm (GA) can efficiently find a solution to a problem in a large space of candidate solutions [21]. It is a search heuristic that mimics the process of natural selection to find the optimal solution, yet in our case the heuristic function will not constantly produce the optimal (or cheapest) solution. Moreover, the design or method of application of GA can also have a significant impact on the quality of the solution [22].

In the following, we extend and adapt GA to our framework to find an acceptable solution in polynomial time.

4.2.1 Security Candidate List

In this paper, we are aiming to find an optimised solution that meets the security requirements while minimising the monetary cost. Therefore, this can be considered a bi-objective optimisation problem. As mentioned in Section 3, each object of the workflow has its security requirements for deploying over clouds. Therefore, we firstly list the satisfied clouds for each object of the workflow in "Candidate List".

The security requirements for each object can be hard constraints (noting it must be met), and the valid clouds that meet these constraints will be maintained in the "Candidate List". Consequently, our problem is reduced to a single objective optimisation which is minimising the monetary cost of the deployment.

ALGORITHM 1: Elitist Prevention

s–the size of elitist list; $elist$–list of elitist individuals; pop–list of all individuals
if $elist$ *is empty* **then**
  ASCsort (pop) ▷ ASCsort sort the pop as ascending order
  $elist \leftarrow$ from $pop[0]$ to $pop[s-1]$
  ▷ copy the first s number of solutions to $elist$
end
for o *in* $\mathcal{O}$ **do**
  for c *in* C **do**
    if o *is securely deployed on* c **then**
      $SArray[o] \leftarrow c$
    else
      $pop \leftarrow$ combine $elist$ and pop
      ASCsort (pop)
      delete s numbers of pop in tail
      $elist \leftarrow$ from $pop[0]$ to $pop[s-1]$
    end
  end
end

ALGORITHM 2: Diversity Protection

pop–a list of all individuals; $size$–size of pop;
$threshold$–predefined threshold of diversity; $rate$–the current mutation rate; $Minrate$–minimum mutation rate;
$Maxrate$–maximum mutation rate.
▷ function $removeDup$ removes the duplications of pop and then copy it to $rpop$
$rpop \leftarrow removeDup(pop)$
$sr \leftarrow$ size of $rpop$
$d \leftarrow 1 - \frac{sr}{size}$
if $d > threshold$ *and* $rate < Maxrate$ **then**
  increase $rate$
end
else if $rate > Minrate$ **then**
  decrease $rate$
end

4.2.2 Elitist Prevention and Diversity Maintenance

The basic GA can be adapted to generate a deployment solution for the problem discussed above. However, to generate an efficient solution, two primary factors: selection pressure and population diversity have to be considered carefully. Moreover, these two factors are inversely related, and so the GA must be carefully designed to balance the effect on the population of diversity and selection pressure. In order to solve this trade-off, we apply the elitism method [23] summarised in Algorithm 1, to increase the selection pressure, and at the same time control the diversity dynamically.

The purpose of the elitist method is to avoid destroying superior individuals in crossover and mutation. Thus, once a solution is confirmed as elitist, it should be directly inherited by the new generation of the population.

Low diversity of a population usually means that the search reaches a local extrema, which significantly impacts the solution. In order to solve this problem, we dynamically control the mutation rate to influence the generation of the new chromosome. Algorithm 2 shows how the diversity protection works.

Firstly, we remove duplications in pop, and count the total number (sr) of unique solutions. Based on that we can have density d of the population, which is represented by the ratio of duplicated solutions. Next, the mutation rate will be increased if the density is greater than the predefined diversity threshold and the mutation rate is less than the maximum mutation rate. If the mutation rate is greater than the maximum mutation rate, it will be decreased.

4.2.3 Adapting to the Framework

The adaption of a genetic algorithm has been divided into five phases: coding, generating a candidate list, initialising individuals, selection, crossover and mutation. We coded our deployment solution as a vector $[s_1^i, s_2^j ....s_n^k]$, where s_i^j means that service s_i is deployed on cloud c_j. In order to reduce the possibility of generating an insecure solution, we chose the clouds from "Candidate List", assigned them to the corresponding objects, and then coded them as above. However, applying these operations will not avoid producing a few new coded solutions which may not meet the security requirements. In such cases, rule SIC is used to verify the security of those solutions. If insecure code is detected, the algorithm will generate a new one to replace it.

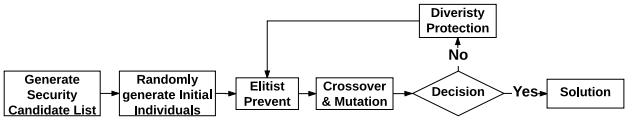


Fig. 3: The overview of adaGA

After generating the initialised individuals, selection, crossover and mutation operations are applied on the individuals to generate new generations. During this process, Algorithm 1 is used to prevent elitist individuals, and two methods are applied to perform the selection: one is a fitness function that can transfer the fitness of a coding into a numeric representation to select superior solutions. For this we can use equation 4 above. The other is a diversity analysis that does not impact the selection result, but influences the crossover and mutation rate.

Crossover then takes part of features from two chromosomes and combines them to generate a new chromosome. The crossover we used is one-point crossover [24].

To enhance the search range, mutation is used. This is implemented by randomly selecting the chromosomes in the current generation and changing them to new secure chromosomes. This operation only happens with very low probability as described in [25]. The overview of adaGA with Elitist Prevent and Diversity Protection Algorithms is shown in Figure 3, the whole processes are repeated until the termination constraints are researched.

In this paper we do not consider how to optimise the parameter setting due to the limited space, however in [26] and [25] authors have explored more justification on parameter setting, such as population size, crossover and mutation probability and stop criteria.Therefore, in our experiment we just follow their techniques about how to set the parameters of GA.

In Section 5 the evaluation of our algorithm will be presented and compared with HUGA (Hybrid Utility-based Genetic Algorithm) [27] which was used to optimise the deployment of workflow over a federated cloud, considering QoS requirements.

4.3 Greedy Algorithm

The greedy algorithm is an iterative algorithm that incrementally finds better solutions. Unlike the Genetic algorithms that need to finish executing before returning a solution, the greedy algorithm generates a valid and improved solution in each iteration. This is a desirable characteristic for systems where the parameters change frequently and the available time for calculating an improved deployment varies significantly.

In this section, we develop a method for finding a deployment solution as an extension of the NCF (Not Cheapest First) [20] algorithm to adapt it to our framework. The NCF is an extended version of greedy algorithm, which uses extra information for planning a deployment, making short term sacrifice for long term benefit. NCF, as summarised in Algorithms 3 and 4, pre-deploys each service on the cloud which minimises the cost and meets the security requirements in isolation, therefore applying a set of optimisation methods to refine the pre-deployment.

The algorithm consists of the following three steps. First, it starts by applying security rules to verify whether security requirements are met by the original workflow. The workflow is valid iff all return values from NRU and NWD are true. Otherwise, the workflow is invalid, the security check returns an error and the whole algorithm stops.

Next, we calculate the cost of deploying services on each valid cloud, using the COD function. COD is calculated by adding the computing cost of service s_i to the transmission cost and storage cost of data sent from all its immediate predecessor services that are not in the same cloud. The initial deployment of services is based on the smallest COD value of each service taking into account the security requirements checked by SIC rule. Function $InitialDeployment$ runs until each service finds a cloud that can meet the security requirement and its smallest COD value associated with this cloud is stored in vector $Temp$.

$$COD(s_i^c) = Scost(s_i^c) + Ccost(s_i^c) + Ecost(s_i^p)$$

Finally, the core idea behind function $Refinement$ is to avoid scheduling services to clouds with huge communication costs. This function includes four cases which detect the services of initial deployment that can be refined. Since the services have been found, these services are assigned to a cloud which minimises deployment costs, while this cloud must meet security rule SIC.

ALGORITHM 3: NCF

W–workflow; S set of service; D–set of dependencies between related services; C–set of available clouds
if *not((WorkflowSecurity(D,S)))* **then**
  Invalid Workflow
else
  INI=InitialDeployment(D,S,C)
end
function **WorkflowSecurity**(D,S)
for $d_{i,j} \in D$ **do**
  if *not* $(NRU(d_{i,j}, s_j)$ *and* $NWD(s_i, d_{i,j}))$ **then**
    return False
  end
end
return True
function **InitialDeployment**(D,C)
▷ Topsort returns a topological order of W
for $s_i \in topsort(\mathcal{W})$ **do**
  for $c_j \in C$ **do**
    if $SIC == True$ **then**
      if $Temp[s_i] > COD(s_i^{c_j})$ **then**
        $Temp[s_i] \leftarrow COD(s_i^{c_j})$
      end
    end
  end
end
return $Temp$

ALGORITHM 4: NCF (Refinement)

$s_{i,max}$– child service of the service s_i with maximum COD value; $SETP(s_i)$– a set which includes service $s_{i,max}$ and all its parent services; $SETC(s_i)$– a set includes s_i and all its child services.
$US = topsort(\mathcal{W})$
for $s_i \in US$) **do**
  switch(s_i)
  case1: $\sum_{s_h \in SETP(s_i)} INI(s_h) > MIN(SETP(s_i))$ **then**
  Deploy all services in $SETP(s_i)$ to the cloud which minimises the cost and $remove(SETP(s_i))$ from US.
  case2: $\sum_{s_h \in SETC(s_i)} INI(s_h) > MIN(SETC(s_i))$ **then**
  Deploy all services in $SETC(s_i)$ to the cloud which minimises the cost and $remove(SETC(s_i))$ from US.
  case3: both case1 and case2 are satisfied **then**
  if $MIN(SETP(s_i) > MIN(SETC(s_i)$ **then**
    Deploy all services in $SETC(s_i)$ to the cloud which minimises the cost $remove(SETC(s_i))$ from US.
  else
    Deploy all services in $SETP(s_i)$ to the cloud which minimises the cost $remove(SETP(s_i))$ from US.
  end
  case4: both case1 and case2 are not satisfied **then**
  Deploy s_i to the cloud which minimises its COD value and s_i remove US
end

4.4 Adaptive Rescheduling Algorithm

In this section, we introduce a heuristic algorithm which has adapted to the DoFCF framework and dynamically generated a new solution for handling cloud availability changes.

The generic adaptive rescheduling algorithm for handling the change of cloud availability works as follows: when a change in cloud availability is detected, the planner estimates the monetary cost for each service based on the available clouds and information (e.g. execution time) of each service. For our case, the estimation heuristic algorithm can be applied to generate a new deployment solution for the services in Selected over available clouds to meet the security requirements as well as minimising the monetary cost. Therefore, services are distributed based on the new solution and then executed. In addition, the execution is monitored until the workflow has been fully executed. Alongside this, a new deployment must be generated in polynomial time, because the time is accounted for in the makespan of the workflow execution, which may bring extra cost. In extreme cases, if it takes considerable time, the available clouds may change again.

We designed and developed a dynamic rescheduling algorithm to be adapted to our framework, which is summarised in Algorithms 5 and 6. In this algorithm, we assume that a workflow W is executed with the deploy-

ALGORITHM 5: Dynamic rescheduling

```
UP– set of unprocessed services; C–set of all available clouds;
DEP_0–deployment
set initial UP = all services of W
while W not finished do
    update C ▷ C is updated via the communication with
    CloudMonitor
    update UP ▷ UP is updated via removing the finished
    services
    if cloud status has changed in C then
        Selected=Redeployservices(UP, C)
        DEP_1 = NCF(Selected, C) if clouds have failed in C
        then
            ▷ submit current execution information to generate
            a new deployment.
            if DEP_1 is not found then
                break;
                ▷ not valid deployment
            else
                submit DEP_1
                DEP_0 ← DEP_1
            end
            ▷ new cloud resources become available
        else
            ▷ if the new deployment is cheaper than the State
            present one
            if DEP_0.cost > DEP_1.cost then
                submit DEP_1
                DEP_0 ← DEP_1
            else
                do nothing
            end
        end
    end
end
```

ment DEP_0. The "Cloud monitor" is used to monitor the workflow execution status and cloud status. Once the monitor has detected any changes in cloud status, function $Redeployservices$ will be invoked to find the services which require redeployment. These actions are summarised in Algorithm 6 as follows. Firstly, the availability of input data for services in set UP is checked, which means that all inputs data must be stored in the available clouds. Otherwise, the elements in set UP will be copied to the $Selected$ set and added to the services which have been completely executed but their outputs are not available to $Selected$. Then, the function tracks back and repeats the same action until the available inputs are found. In this case, $Selected$ will include the services in UP as well as the added services. Otherwise, $Selected$ is equal to UP.

Based on the selected services and the available clouds, a new deployment DEP_1 can be generated by applying a NCF algorithm. If the cloud change is caused by the failure of some clouds, DEP_1 is used to deploy the services in $Selected$ directly. Otherwise, if the change is caused by a new cloud becoming available, then the cost of DEP_1 and DEP_0 will be compared (only considering the unfinished services). If DEP_1 is cheaper, the workflow will be scheduled to a new deployment solution, otherwise the execution status and cloud status are monitored until the workflow is completely executed.

4.5 Time Complexity

The time complexity of each algorithm is formalised as follows:

ALGORITHM 6: Redeploy Services

```
DEP_0–the original deployment; Selected-set of services for
redeploying; parents(s_i)–set of s_i's parent services
function Redeployservices(UP,C)
Selected ← UP for s_i ∈ UP do
    FINDNODES(parents(s_i), C, Selected)
end
return Selected
function Findnodes(parents(s_i),C,Selected)
if parents(s_i) == ∅ then
    return
else
    for s_j ∈ parents(s_i) do
        if s_j ∉ UP then
            ▷ DEP_0(s_j) indicates the cloud on which s_j was
            deployed
            if DEP_0(s_j) ∈ C then
            else
                add s_j to Selected if not included
                FINDNODES(parents(s_j), C, Selected)
            end
        end
    end
end
```

HUGA algorithm can be split into three phases in order to analyse the time complexity: selection, crossover and mutation. Therefore, the time complexity of the selection phase is $O(|P| \times |G| \times |\mathcal{O}|)$ where P is the size of population and G is the number of generations. For crossover and mutation phases, we need to operate on the whole chromosome, so the complexity of both is $O(|P| \times |G|)$. Thus the time complexity of HUGA is $O(|P| \times |G| \times |\mathcal{O}|)$.

Comparing *adaGA* with *HuGA*, adaGA contains an additional Elitist phase for each generation which slightly increases the time complexity. In each Elitist phase, the algorithm observes the chromosomes for each population and saves the best one. Thus, the complexity of this phase is $O(|P| \times |G| \times |\mathcal{O}|)$. Therefore, the time complexity of *adaGa* is $O(|P| \times |G| \times |\mathcal{O}|)$ as well.

In the *NCF* algorithm, we have already analysed the time complexity in paper [20]. It is significantly impact by the structure of the workflow. if the workflow is linear, the complexity in the worst case becomes $O(|\mathcal{O}| \times |C|)$. Conversely, for a star-shaped workflow the best case complexity is $O(|D| \times |C|)$.

5 Experiments and Evaluation

To evaluate the performance of our proposed framework, we conducted a series of simulation experiments on a number of real scientific workflow applications. In addition, we applied our framework to dynamically deploy a scientific workflow over a set of e-SC instances on multiple clouds. The experimental setting, and results are presented in the following subsections.

5.1 Simulation Environment

In this work, the experiments must be repeatable in order to easily compare and analyse different types of algorithms. Therefore, to ensure repeatability, we evaluated our framework using CloudSim to investigate the deployment and scheduling of workflows in federated cloud environments.

Workflow	Medium	Large	Very large
CyberShake	30	100	1000
Montage	25	100	1000
LIGO	30	100	1000
Epigenomics	24	100	995

TABLE 2: Number of Tasks of each Workflow for Each of the Three Scales

Type	Location	Exec (/hour)	Store (/hour/GB)	In (/GB)	Out (/GB)
C1	0	0.40	0.10	0	0.02
C2	2	2.20	0.60	0.03	0.01
C3	1	1.23	0.30	0.14	0.07
C4	2	3.70	0.60	0.10	0.05
C5	3	4.50	0.90	0.14	0.05
C6	4	5.5	1.30	0.14	0.13

TABLE 3: Cloud Pricing and Security Levels

5.1.1 Experiment Setup

In the evaluation of our framework, we consider four common types of workflow applications: CyberShake (earthquake risk characterisation), Montage (generation of sky mosaics), LIGO (detection of gravitational waves) and Epigenomics (bioinformatics).[1] The full characterisation of these workflow applications can be found in [28], however, we only consider the execution time, and the input and output data of each service. Table 2 lists the four workflow types with different numbers of tasks: medium, large and very large.

The data privacy information for the workflows is unavailable to be used for assigning the security levels of each object. Therefore, we randomly generated the security levels for each object in these workflows.

Six VMs have been created, representing workflow execution environments, in six different data centres to represent six types of cloud (with different security levels). Additionally, each VM can run several services at the same time. In this paper, we do not consider the performance issue. Therefore each cloud shares the same configuration, within 1 core, 2 GB RAM and 12GB Disk. Table 3 details number of clouds with location security levels, computation cost, storage cost and communications cost, where In and Out represent the transferring cost of incoming and outgoing data respectively.

The experiment results, presented below, are the average values of observing 1000 executions of each algorithm for each type of workflow. For each of the 1000 repetitions, the same random number generation seeds for each execution, which guarantees each algorithm is exactly running over the same infrastructure.

5.1.2 Monetary Cost Evaluation

As discussed earlier, the total cost includes execution cost, running time storage cost and communication cost. The pricing of each cloud listed in Table 3 shows that a more secure cloud is generally more expensive. Our cost calculation does not consider additional costs like license and VM image costs which are charged by some cloud providers.

1. The XML description files of the workflows are available via the Pegasus project: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

Figs 4, 5 and 6 depict the normalised cost for the four types and three size of workflow applications mentioned previously using three algorithms: NCF, HUGA, and adaGA. Each figure represents cost calculations for a specific workflow size to show the variations in the cost according to the size.

The results show that the algorithm adaGA can always generate the cheapest deployment solution. For example, in a case with medium size of "Montage" workflow, the solution generated by adaGA can save up to 35% compared to the NCF solution.

The types of workflow significantly impacted the solutions generated by NCF. {*Fig 4 shows that the costs of the deployment solutions generated by the three algorithms are very close when these algorithm are applied to the workflows of the LIGO and Epigenomics in Medium size.* } However, for the other two types of workflows, the solutions generated by NCF are much more costly than adaGA. Furthermore, the differences are reduced with the increase in workflow size. This is because NCF is not influenced by the search space (larger workflow indicates more deployment solutions). In addition, the search space significantly impacted the results generated by adaGA and HUGA. However, the Elitist Prevent and Diversity Protection methods were used in adaGA to avoid the algorithm visiting the less desirable solutions.

5.1.3 Time Complexity Evaluation

In order to evaluate the time complexity of each algorithm, we measured the time consumed by each to find the optimised deployment for the four types of workflow on the given clouds. According to our evaluation, Fig 8 shows that algorithm NCF is significantly faster than the other algorithms. Further, adaGA has better performance than HUGA with medium size workflows. The reason is that the search will be terminated if no better solution has been found after repeating the pre-defined generations. Nevertheless, as the workflow size increases, adaGA consumes more time than HUGA to find the deployment solution. However, the deployment solution for a very large size of "CyberShake" workflow can be generated by adaGA in less than one minute. Consequently, by considering cost savings, adaGA can be the better choice.

Fig 8 also indicates that the time complexity of each algorithm is not only dependent on the number of o (data and services) and the available clouds, but also on the structure of the target workflows and the security levels for each o and cloud.

5.1.4 Cloud Availability Change Evaluation

In this part of the experiments, we simulated the change in cloud availability by predefining the times when each cloud was available. To do this we set the start time and termination time for each cloud before starting the simulation. A *Cloud Monitor* was implemented to monitor cloud status, i.e. detect changes in cloud environments. If a changed status is detected, a notification is sent to the *broker*. Thus, the broker can reschedule the running workflow to the available clouds based on the cost of the new deployment. This was evaluated for two types of change: *Clouds fail* and *Availability of new clouds*.

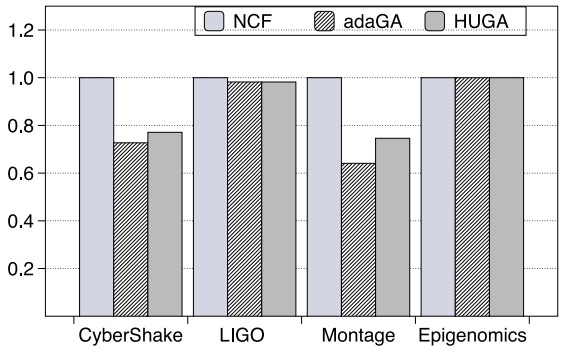


Fig. 4: Cost for Medium Size Workflows

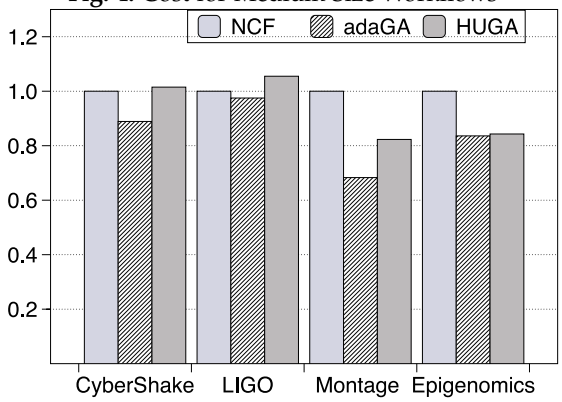


Fig. 5: Cost for Large Size Workflows

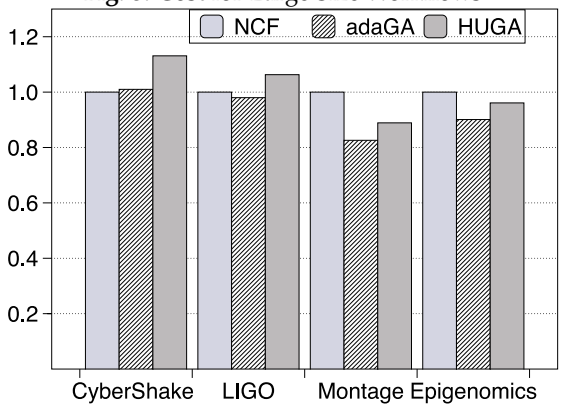


Fig. 6: Cost for Very Large Size Workflows

Fig. 7: The observed outputs metrics are execution cost normalised with the correspondent value obtained from the NCF algorithm, noting $Others/NCF$

Cloud fail was simulated by setting the terminal time for the randomly selected clouds as uniformly distributed between 0 and the $makespan$. In another words, during the workflow execution, the number of cloud failures is randomly between 0 to 6. Furthermore, we performed 1000 simulations, each with different cloud failure settings, and recorded each execution, including how many clouds fail, the makespan, completion of workflow execution. Consequently, we can have the average of the cost and the makespan based on the recording.

In this evaluation, we used "Epigenomic " workflow with medium, large and very large size. As an example of the setting for the medium workflow size, we randomly selected the clouds and set their termination time as uniformly distributed over $[0, makespan]$ where the $makespan$ is the execution time of the selected workflow running as the deployment generated by NCF. For the clouds that were not selected, the termination time was set as infinite.

Thus, we can have three types of executions: (1) Success: in this type, the workflow is completely executed without rescheduling as the selected clouds finished executing in the assigned tasks before their terminating time. (2) SBR: the workflow is successfully executed by rescheduling on the currently available clouds after a cloud has become unavailable. (3) Fail: the workflow execution cannot be completed as no alternative deployments were available after a cloud failure. This would be because of cloud unavailability or because the available clouds do not meet the security requirements.

Table 4 depicts the results of cloud failure. These demonstrate that the failures occur more frequently with the increasing sizes of workflow and execution time. We normalise the results by using the ratio of value of the outputs of the SBR executions with that from the Success executions, recording $SBR/Success$. In the case of a very large workflow, we only have to pay an extra 2% money and 33% time to avoid re-running the workflow from the beginning when a failure happens during the workflow execution.

Workflows	Execution status (%)	Cost	Time
Medium	Success:74	1	1
	SBR:16	1.14	2.36
	Fail:10	none	none
Large	Success: 70	1	1
	SBR: 20	1.1	1.15
	Fail:10	none	none
Very Large	Success: 54	1	1
	SBR: 22	1.02	1.33
	Fail: 24	none	none

TABLE 4: Experiment results for cloud failure

To simulate *Availability of new clouds*, the workflow has to be already running over a set of clouds with a deployment. Thus, we pre-set clouds C6 and C5 as available for the initial deployment. Furthermore, we randomly selected clouds and set the start time of them as uniformly distributed over $[0, \infty]$. We generated 1000 settings and repeated the executions of the rescheduled workflow to new clouds based on the monetary cost savings. This resulted in the execution types shown in Table 5: (1) *Success* represents no new clouds becoming available or the new clouds is not offering cheaper deployment than the currently running one. (2) *SBR* denotes the running workflow can be rescheduled to the new available clouds to save execution costs.

Table 5 shows that the fluctuations of the saving cost are very significant, depending on the types of the workflow and available clouds. (In Table 5, we used the same normalisation rule as Table 4). Therefore, when new clouds become available, users should participate in making the decision to use a new deployment based on the estimated saving cost and makespan.

Moreover, Table 5 shows that the makespan goes up, while the monetary cost goes down. The reason is that when a running task is shifted to a new cloud which is cheaper, the running task has to be killed, and then be redeployed and re-executed in the new cloud. If the new cloud is faster than

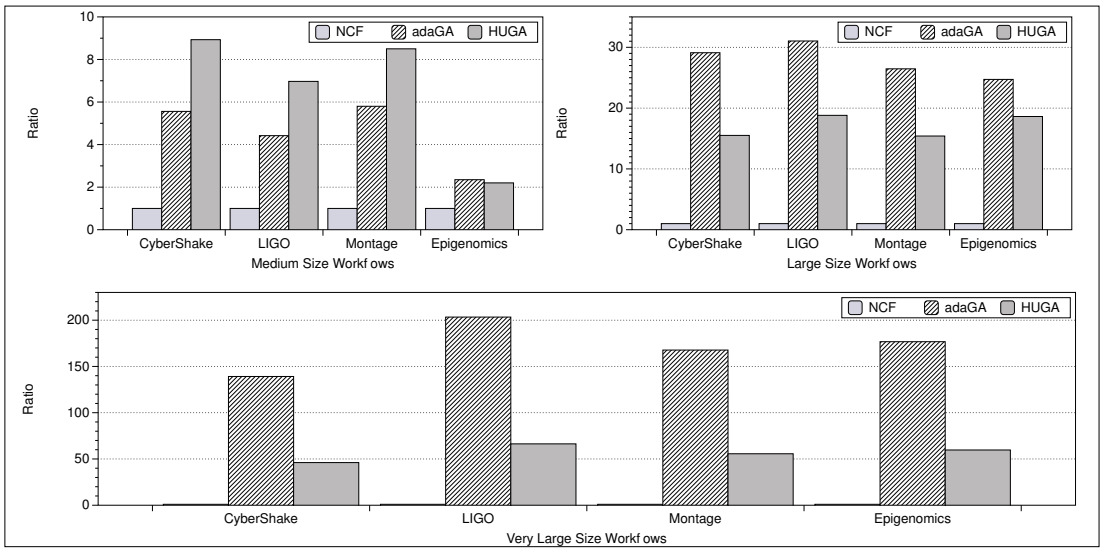


Fig. 8: The observed outputs metrics are execution time normalised with the correspondent value obtained from the NCF algorithm, noting $Others/NCF$

the current host cloud, the makespan might be reduced. However, in this work we do not consider the performance of the clouds and assume that clouds' performance are the same.

Workflows	Execution status (%)	Cost	Time
Medium	Success:92 SBR:8	1 0.98	1 1.46
Large	Success: 88 SBR: 12	1 0.90	1 1.21
Very Large	Success: 70 SBR: 30	1 0.99	1 1.33

TABLE 5: Experiment results for new cloud available

5.2 Realistic System Evaluation

To evaluate our algorithm in conditions closer to a production use, we applied it to schedule scientific workflows in e-Science Central(e-SC). We used the e-SC APIs to create a Cloud Services Broker that can orchestrate invocations of a single workflow partition over a number of e-SC instances.

5.2.1 Design and Setup

According to the architecture of the cloud services broker, shown in Fig 1, our tool consists of three components: Client, Cloud Services Broker and Federated Cloud.

The *Client* includes a user interface (UI) which allows users to create workflows for the e-Science Central workflow engine. The description of the created workflow can then be passed to the Broker. *Cloud Services Broker* is the core part of our tool and includes a planner to assign workflow to federated cloud using the algorithms discussed earlier. e-SC APIs are used to dispatch tasks to corresponding clouds and monitor the execution. Failure Generator is used for simulating failures by turning on or shutting down e-SC instances. *Federated Cloud* is a set of e-SC instances which can interact with the broker and other e-SC instances through e-SC APIs, and process the tasks which are scheduled.

To evaluate our tool we selected one of the workflows used in the cloud e-Genome project [29].

The workflow was implemented to process exome sequenced by using e-SC deployed on Microsoft Azure cloud.

While in the e-Genome project security aspects are not a primary concern, guaranteeing that human genomic data can be securely processed on the cloud is very important. Therefore, we modelled the security requirements of the selected e-Genome workflow by assigning security levels as shown in Tables 6 and 7. Note that the data size transferred among blocks and the execution time of each block are real values taken from logs collected by e-SC. Table 6 shows data sizes in GB, where 0 denotes less than 1 MB of data. The pricing of Clouds C1, C2 and C3 in Table 3 was applied to calculate the deployment cost.

To simulate this environment we set up three virtual machines, each running a single instance of e-SC system. VM1 was hosted on a personal PC and represented the private cloud. Two other VMs were hosted in our University virtualised environment and played the role of public cloud providers C2 and C3.

Service	Name	Clearance	Location	Time(/h)
$Sample\_Name$	S1	1	0	1
$ImportFile$	S2	1	0	1.5
$Sample\_Value$ Info	S3	1	0	3
$HG19$	S4	1	0	0.1
$Filter$	S5	2	0	10
$Exome-Regions$	S6	1	0	7
$Intervalpadding$	S7	0	0	20
$ColumnJoin$	S8	2	0	0.1
$AnnotateSample$	S9	2	0	5
$Export$	S10	1	0	0.3

TABLE 6: Services representation and security and execution time

Our evaluations included three steps: the first step tests the static deployment algorithm. We kept all three e-SC instances running and applied adaGA to make the deployment plan. The second step shows how to handle a cloud failing, by shutting down one of the running e-SC instances when the workflow was running. The setting of this step is similar to that of CloudSim. Finally, we tested the avail-

Data	Location	Size (GB)
$S_{1,8}$	1	0
$S_{2,5}$	0	1.1
$S_{3,8}$	2	0.01
$S_{4,5}$	0	0.005
$S_{4,7}$	0	0.005
$S_{5,7}$	0	6.2
$S_{6,7}$	0	10.3
$S_{7,9}$	1	3.6
$S_{8,9}$	0	0
$S_{9,10}$	0	0.05

TABLE 7: Data security and size

ability of a new cloud by deploying the given workflow on two clouds, and then turning on a new instance which offers price advantage. Also, for the purpose of the experiment we reduced the execution time of the given workflow to about 30 seconds by scaling down the amount of input data shown in Table 7 by a factor of 6000.

5.2.2 Results and Analysis

Based on the presented experiment setup, all of the three steps of the deployments are illustrated in Table 8. Precisely, "Static" refers to first step, and "Cloud fail" and "New Cloud" correspond to steps two and three respectively.

Static, shown in Table 8, represents the cheapest solution which was generated using the adaGA algorithm by deploying the workflow over cloud C1, C2 and C3 to meet the security requirements. Services S_1 S_7 and S_8 were deployed on C2 and others were allocated on C1.

For the Cloud Fail, we used the deployment of *Static* as the initial deployment (INI). However, cloud C1 failed (shown as blue in Table 8) when service S_9 was ready to execute.

The available clouds are C2 and C3, and the inputs of S_9 are stored in C2 (the outputs of S_7 and S_8), therefore, S_9 can be rescheduled to C3 to continue the execution (indicated in "Cloud fail", SBR). If S_7 is deployed on C1 with the same failure, S_2, S_4, S_5, S_6 and S_7 should be re-executed in C3. Thus S_9 and S_{10} can be completed in C3.

In the third step, C2 and C3 were available for the initial deployment (see Table 8 New Cloud, INI). After the workflow was executed for one second , C1 became available. S_1 and S_4 were completely executed on C2 and C3 respectively, but S_2, S_6 and S_3 were still running. Based on the status information, a cheaper deployment solution (see New Cloud SBR in Table 8) became available, which required termination of S_2 and S_6, and then re-running them on C1, as shown in green in Table 8.

Service	Static	Cloud fail		New Cloud	
		INI	SBR	INI	SBR
S_1	C2	C2	C2	C2	C2
S_2	C1	C1	C1	C3	C1
S_3	C2	C2	C2	C2	C2
S_4	C1	C1	C1	C3	C3
S_5	C1	C1	C1	C3	C1
S_6	C1	C1	C1	C3	C1
S_7	C2	C2	C2	C2	C2
S_8	C2	C2	C2	C2	C2
S_9	C1	C1	C3	C3	C1
S_{10}	C1	C1	C3	C3	C1

TABLE 8: Two deployments

Table 9 shows average values of the cost and makespan of each deployment by repeating the executions 10 times. Where SBRF represents the situation of handling C1 fail (see Table 8 Cloud Fail SBR). Similarly, ININ and SBRN are the experimental results of new cloud available, where the makespan of SBRN is approximate one second more than others (it will take one hour more by using the original inputs). It is because C1 has to re-execute S_2 and S_6 from the beginning.

Deployment	Time (seconds)	Cost
Static	28.93	64.44
SBRF	28.94	67.23
ININ	28.92	84.62
SBRN	29.90	65.17

TABLE 9: The cost of different deployments

6 Related Work

Cloud computing is a technology for transforming computing as utility model such as water, electricity, gas and telephony [30]. Therefore, it is unlike grid in that the total ownership cost of running a workflow is considered to be a much more important optimisation criterion. Compared with other computing resources, cloud has unique features: pay-as-you-go pricing, multiple instance types, elasticity, without operation of infrastructure and so on. Thus, the state-of-the-art techniques or mechanisms for workflow management need to be adapted to the new computing environments.

On a single cloud, most research efforts are aimed at improving the performance of workflow systems. In [31], the authors introduced an auto-scaling method that applied a fixed sequence of transformations on the workflow in a heuristic way. However, the sequence of workflow transformations are not unique, and different transformations have quite different costs.

The most common approach is concentrated on workflow scheduling for running workflow in cloud to meet performance and cost constraints. The authors in [32] described a method which can dynamically provision VMs for meeting the performance requirements of executing workflows, and recover the computing resources when they are over provisioned to reduce the monetary cost. Kllapi et al. presented a method to deal with the trade-off between makespan and financial cost for data processing flows [33]. The authors in [11] proposed an algorithm that uses the idle time of provisioned resources and surplus budget to replicate tasks so as to increase the likelihood of meeting deadlines. At the same time, the economic cost of execution is also minimised by carefully planning the provision of VMs.

Considering security-driven scheduling, only few groups of researchers have investigated this topic from different angles in various contexts. Mace et al. [34] explored the current information security issue of public cloud and provided general security solutions to choose what workflows or subsets of workflows can be executed in a public cloud while ensuring the security requirements are met. The authors in [35] proposed a security-driven scheduling algorithm for DAGs workflow which can achieve high quality of application security, based on task priority rank to estimate

the security overhead of tasks. In addition, this algorithm considered improving the performance of workflow execution.

Cloud federation work is relatively new in the cloud computing area. Therefore, there is little available literature. In [36], the private cloud (the user's own machines) was also assumed to be a free computing resource, with limited computing power. A public cloud such as Amazon EC2 can meet users' performance requirements, but the cost must also be minimised. A framework, called PANDA (PAreto Near optimal Deterministic Approximation), was designed for scheduling workflow across both public and private clouds with the best trade-off between performance; hence Pareto-optimality. Fard et al in [37] introduced a pricing model and truthful mechanism for scheduling workflow to different clouds, considering the monetary cost and completion time. In order to solve the trade-off between cost and performance, a Pareto-optimal solution is adapted in the scheduling algorithm. However, none of them considered security and cloud availability change.

SABA (Security-Aware and Budget-Aware workflow scheduling strategy) [9] provided a static workflow deployment solution over multi-clouds for optimising security, makespan and monetary cost. The optimisation in this work was based on a heuristic list which ranks the priority of each task of the workflow through a normalisation function. Jrad et al. [27] proposed a cloud broker to schedule larger scientific workflows over federated clouds to match the QoS and cost requirements. Since these two efforts are static scheduling algorithms, they are unable to manage the cloud availability change.

Regarding exception handling, the data flow-based approach in [38] was introduced to support hierarchical exception propagation and user-defined exception. This work considered the exceptions caused by workflow itself, while we focused on solving computing resource change problems.

Furthermore, in our previous work [10], we proposed a dynamic method to handle cloud failure issues, while the workflow is running on a federated cloud. However, this method can not handle large scale workflow and also did not consider the situation that a new cloud becomes available.

7 Conclusion and Future Work

In this paper, we have presented the DoFCF framework to improve the security of the workflow applications while they are distributed on a federated cloud. A cost model has been designed to optimise the cost of each deployment option. Furthermore, we developed a novel dynamic rescheduling method and added it to handle the change of cloud resources availability in our framework. This will support execution resuming when clouds fail and save the cost when new clouds become available. Additionally, we designed and implemented two algorithms for static deployment planning, i.e., NCF and adaGA. These algorithms have been applied to different types of workflows, then their performance was discussed and analysed.

We evaluated the performance of our developed framework by conducting a series of experiments on various types of real scientific workflows. The experiments have been performed using a simulation environment as well as real workflow management system. The results show that our framework is suitable for deploying universal scientific workflows over federated clouds.

As future work, we will develop a matrix to measure the security level of cloud datacenter. The existing studies only consider the risks of cloud computing, but none of them provides a quantitative measurement. This measurement can be a strong support for this paper by providing the realistic security level of each cloud datacenter.

Acknowledgments

This work was supported by the RCUK Digital Economy Theme [grant number EP/G066019/1 - SIDE: Social Inclusion through the Digital Economy], and the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 600854 SmartSociety Hybrid and Diversity-Aware Collective Adaptive Systems: Where people meet machines to build smarter societies (http://www.smart-society-project.eu/).

References

[1] G. Juve and E. Deelman, "Scientific workflows in the cloud," in *Grids, Clouds and Virtualization*. Springer, 2011, pp. 71–91.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. (2009) Above the clouds: A berkeley view of cloud computing.

[3] F. Zhang and M. Sakr, "Performance variations in resource scaling for mapreduce applications on private and public clouds," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 456–465.

[4] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 164–177, April 2012.

[5] Z. Zheng, H. Ma, M. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *Services Computing, IEEE Transactions on*, vol. 6, no. 3, pp. 289–299, July 2013.

[6] K. Finley. Godaddy outage takes down millions of sites, anonymous member claims responsibility, year = 2012, url = http://techcrunch.com/2012/09/10/godaddy-outage-takes-down-millions- of-sites/, urldate = 2012.

[7] (2011) Summary of the amazon ec2 and amazon rds service disruption in the us east region. [Online]. Available: http://aws.amazon.com/message/65648/

[8] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ser. ICA3PP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 13–31.

[9] L. Zeng, B. Veeravalli, and X. Li, "Saba: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of Parallel and Distributed Computing*, vol. 75, no. 0, pp. 141 – 151, 2015.

[10] Z. Wen and P. Watson, "Dynamic exception handling for partitioned workflow on federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec 2013, pp. 198–205.

[11] R. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1787–1796, July 2014.

[12] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-science central platform," *Royal Society of London. Philosophical Transactions A. Mathematical, Physical and Engineering Sciences*, vol. 371, p. 20120085, 2013.

[13] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.
[14] "Jclouds, howpublished = https://jclouds.apache.org, note = Accessed: 2015-09-24."
[15] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.
[16] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 180–188.
[17] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations," MITRE Corporation, Tech. Rep., Mar. 1973.
[18] C.E.Landwehr, "Formal models for computer security," *ACM Computing Surveys*, vol. 13, 1981.
[19] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin, Heidelberg: Springer-Verlag, 1994.
[20] Z. Wen, J. Cała, and P. Watson, "A scalable method for partitioning workflows with security requirements over federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, Dec 2014.
[21] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[22] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 5, pp. 56–60, 2012.
[23] D. Bhandari, C. Murthy, and S. K. Pal, "Genetic algorithm with elitist model and its convergence," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 10, no. 06, pp. 731–747, 1996.
[24] R. Poli and W. B. Langdon, "Genetic programming with one-point crossover and point mutation," in *Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag, 1997, pp. 180–189.
[25] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, p. 102, 1989.
[26] W. Sun, "Population size modeling for ga in time-critical task scheduling," *International Journal of Foundations of Computer Science*, vol. 22, no. 03, pp. 603–620, 2011.
[27] F. Jrad, J. Tao, I. Brandic, and A. Streit, "{SLA} enactment for large-scale healthcare workflows on multi-cloud," *Future Generation Computer Systems*, vol. 4344, no. 0, pp. 135 – 148, 2015.
[28] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682 – 692, 2013, special Section: Recent Developments in High Performance Computing and Security.
[29] J. Cała, Y. X. Xu, E. A. Wijaya, and P. Missier, "From scripted HPC-based NGS pipelines to workflows on the cloud," in *Procs. C4Bio workshop, co-located with the 2014 CCGrid conference*. Chicago, IL: IEEE, 2014.
[30] D. Williams, H. Jamjoom, and H. Weatherspoon, "Plug into the supercloud," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 28–34, March 2013.
[31] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, Nov 2011, pp. 1–12.
[32] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 22:1–22:11.
[33] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. Ioannidis, "Schedule optimization for data processing flows on the cloud," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 289–300.
[34] J. Mace, A. van Moorsel, and P. Watson, "The case for dynamic security solutions in public cloud workflow deployments," in *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, June 2011, pp. 111–116.
[35] T. Xiaoyong, K. Li, Z. Zeng, and B. Veeravalli, "A novel security-driven scheduling algorithm for precedence-constrained tasks in heterogeneous distributed systems," *IEEE Trans. Comput.*, vol. 60, no. 7, pp. 1017–1029, Jul. 2011.
[36] M. HoseinyFarahabady, Y. C. Lee, and A. Zomaya, "Pareto-optimal cloud bursting," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 10, pp. 2670–2682, Oct 2014.
[37] H. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1203–1212, June 2013.
[38] X. Fei and S. Lu, "A dataflow-based scientific workflow composition framework," *Services Computing, IEEE Transactions on*, vol. 5, no. 1, pp. 45–58, Jan 2012.

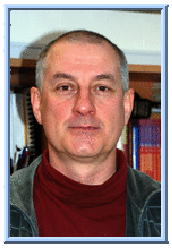
Zhenyu Wen received the B.E. degree in computer science and technology from Zhejiang Gongshang University, Zhejiang, China, in 2009, and the M.S and Ph.D. degree in computer science from Newcastle University, Newcastle Upon Tyne, U.K., in 2011 and 2015. He is currently a PostDoctoral Researcher with the School of Informatics, the University of Edinburgh, Edinburgh, U.K. He has authored a number of research papers in the field of cloud computing. His current research interests include Multi-objects optimisation, Crowdsources, Artificial Intelligent and Cloud computing.

Rawaa Qasha is a 3rd year PhD student at the school of Computing Science, Newcastle University, UK. I received the master degree from Computer Sciences department, University of Mosul, in 2000. Prior to start PhD career I was a lecturer (assistant professor) in computer science at University of Mosul, Iraq. My research interests concentrate on Cloud computing, distributed system, workflow deployment, E-Science system.

Zequn Li received the B.E degree in Computer Science from Shandong University of Finance and Economics in Shandong, China and M.S degree in Advanced Computer Science from Newcastle University in Newcastle upon Tyne, United Kingdom. Currently he is a PhD student in the school of Mathematics and Information Science, Northumbria University, United Kingdom. His research interests include Machine Learning and Distributed Systems.

Rajiv Ranjan Dr. Rajiv Ranjan is a Associate Professor (Reader) in Computing Science at Newcastle University, United Kingdom. At Newcastle University he is working on projects related to emerging areas in parallel and distributed systems (Cloud Computing, Internet of Things, and Big Data). Previously, he was Julius Fellow (2013-2015), Senior Research Scientist (equivalent to Senior Lecturer in Australian/UK University Grading System) and Project Leader in the Digital Productivity and Services Flagship of Commonwealth Scientific and Industrial Research Organization (CSIRO Australian Governments Premier Research Agency). Prior to that he was a Senior Research Associate (Lecturer level B) in the School of Computer Science and Engineering, University of New South Wales (UNSW). Dr. Ranjan has a PhD (2009) in Computer Science and Software Engineering from the University of Melbourne. Dr. Ranjan is broadly interested in the emerging areas of distributed systems. The main goal of his current research is to advance the fundamental understanding and state of the art of provisioning and delivery of application services (web, big data analytics, content delivery networks, and scientific workflows) in large, heterogeneous, uncertain, and emerging distributed systems.

Paul Watson is Professor of Computer Science, Director of the Informatics Research Institute, and Director of the North East Regional e-Science Centre. He also directs the UKRC Digital Economy Hub on Inclusion through the Digital Economy. Prior to moving to Newcastle University, Paul worked at ICL as a system designer of the Goldrush MegaServer parallel database server. He previously gained a Ph.D. and became a Lecturer at Manchester University. His research interests are in scalable information management including data-intensive e-Science, dynamic service deployment and e-Science applications. In total, Paul Watson has over forty refereed publications, and three patents.



Alexander Romanovsky is a Professor with the School of Computing Science (Newcastle University, UK) and the leader of the School's Secure and Resilient Systems Group. He coordinates the EPSRC platform grant on Trustworthy Ambient Systems and the RSSB SafeCap+ grant on railway integrated optimum capacity, safety and energy strategies. He is a co-investigator of the EPSRC PRiME programme grant on Power-efficient, Reliable, Many-core Embedded systems. Before this, he coordinated the major EU FP7 DEPLOY IP that developed the Rodin tooling environment for formal stepwise design of complex dependable systems using Event-B. Rodin is now widely used by companies in Europe, China, Japan, USA, Canada and Brazil. Prof Romanovsky's main research areas are system dependability, fault tolerance, safety, modelling and verification.